Bayes Net Inference
Swapnil Mishra - u5053816

COMP6470 - Special Topics in Computing

Supervisor: Dr Scott Sanner

November 1, 2013
Abstract

Bayesian networks are used extensively in field of graphical models as tool for representing and working on various real world problems. Here we present a study of an exact inference algorithm, named Variable Elimination, and an approximate inference algorithm, named Gibbs Sampling. The analysis of algorithms concludes, although Gibbs sampling is not exact but for large networks it yields results with an tolerable error in accepted amount of time.
Acknowledgments

First of all I would like to thank my supervisor Dr Scott Sanner for providing his constant guidance, motivation and time for his discussions and talks. I cannot thank my family and friends enough for being a constant source of help and courage.
Contents

1 Graphical Models 3
  1.1 Overview .................................................. 3
  1.2 Bayesian networks ....................................... 3

2 Inference 5
  2.1 Variable Elimination(VE) ................................. 6
  2.2 Gibbs Sampling ........................................... 8

3 Experiments and Results 9
  3.1 Setup ...................................................... 9
  3.2 Results .................................................... 9
  3.2.1 Time Analysis ........................................ 9
  3.2.2 Error Analysis ......................................... 11
  3.2.3 Mixing Time ........................................... 12

4 Conclusion 15
Chapter 1

Graphical Models

1.1 Overview

Graphical models are probabilistic network used to represent real world problems of various domains. They provide an easy way to represent conditional probabilities between various random variables and facilitate efficient computation of algorithms over those tables. There are two main representations of graphical models, Bayesian networks (BNs) and Markov random fields (MRFs). Throughout this paper we will consider BNs as our preferred representation.

1.2 Bayesian networks

Bayesian networks are directed graphical models. They are directed acyclic graphs (DAG) representing dependency relationship between various random variables. The nodes of the graph represent a distinct discrete random variable. The edge from a node to other node represent the direct dependencies between them and absence signifies conditional independence between them. For example, consider a network with three random variables $x_1$ to $x_3$. Then the joint probability of this distribution is given by:

$$p(x_1, x_2, x_3) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)$$

Now as per BN in figure 1.1, we have $x_1$ and $x_2$ being conditionally independent of each other as they are not connected. Hence we can rewrite the
joint distribution as:

\[ p(x_1, x_2, x_3) = p(x_1)p(x_2)p(x_3|x_1, x_2) \]

The joint distribution defined by a BN graph is given by the product, over all of the nodes of the graph, of a conditional distribution for each node conditioned on the variables corresponding to the parents of the node in the graph. Let \( x = x_1 \ldots x_K \) and \( pa(x_k) \) denote the set of parents of \( x_k \) then the joint distribution is given by [1]:

\[ p(x) = \prod_{k=1}^{K} p(x_k|pa(x_k)) \]

The two conditional independence properties characterised by a BN are:

- Each random variable is conditionally independent of its non-descendants given it parent variables.

- Each random variable is conditionally independent of all other variables in the network given its Markov blanket i.e. its parents, its children and all other parents of its children.
Chapter 2

Inference

The most fundamental task in any probabilistic network is to calculate the posterior probability distribution over a set of query variables given a certain evidence. This task is known as inference for BN.

The task we are interested in finding is $p(Q = q|E = e)$ where $Q = q$ is the query variable and $E = e$ is the given known condition. Using conditional probability rules we know that:

$$p(Q = q|E = e) = \frac{p(Q = q, E = e)}{p(E = e)} \quad (2.1)$$

Now it is very simple to find $p(q, e)$ as it can be very well calculated by summing out all the values from the joint distribution where $(q, e)$ holds, it is given by:

$$p(q, e) = \sum_{x \neq q, e} p(q, e, x - q - e) \quad (2.2)$$

And similarly we can find $p(e)$ by summing out the joint probability above. It can be written as:

$$p(e) = \sum_{q} p(q, e) \quad (2.3)$$

Hence we can compute inference for any random variable given if we have the whole table representing joint distribution. As we can see from above, if we have $V$ different random variables each with $K$ different states, the time complexity of inference would be $O(K^V)$. As it is very expensive we use exact inference and approximate inference for feasible time complexity [1].
2.1 Variable Elimination (VE)

Let us assume we have the graph given as in figure 2.1, where all random values are boolean. To compute $p(x_4)$ we use the of BN to get:

$$p(x_4) = \sum_{x_3} \sum_{x_2} \sum_{x_1} p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_3)$$  \hspace{1cm} (2.4)

As per equation 2.4 we will need 48 multiplications and 14 additions to perform this calculation. If we push in the summation inside, we can rewrite the equation as:

$$p(x_4) = \sum_{x_3} p(x_4|x_3) \sum_{x_2} p(x_3|x_2) \sum_{x_1} p(x_1)p(x_2|x_1)$$  \hspace{1cm} (2.5)

The number of operations required for calculations done in equation 2.5 is 18 (4 multiplications and 2 additions at each level). This reduction is the main idea behind VE. We can summarize the ideas that helps us to prevent blowup of joint distributions are:

- Structure of BN makes sure that some joint expressions are conditioned on only few random variables.
- Computing of results and then caching them, helps to reduce their generation exponentially again and again.

Any marginal probability computation can be seen as taking product of all conditional probabilities distributions and then summing over all random variables apart from query variable. This can be done in any order we want, till we make sure that summation over a random variable is performed only after multiplying all factors it is involved in. Thus in general the task is to calculate $\sum_{\mathbf{Z}} \prod_{\phi \in \Phi} \phi$ where $\Phi$ is a set of factors on random variables of network and $\mathbf{Z}$ is set of all random variables to be marginalized. VE eliminates one variable $Z_i$ at a time. It finds the set factors that have $Z_i$ in them and then multiplies them together. Once multiplication is done it it marginalizes $Z_i$ and replaces the old factor with this new factor [1], refer Algorithm 1.
Algorithm 1 Sum-product variable elimination algorithm [1]

function SUM-PRODUCT-VE (Φ, Z, ≺)
▷ Φ Set of factors ▷ Z Set of variables to be eliminated ▷ ≺ Ordering on Z
Let $Z_1, \ldots, Z_k$ be an ordering of $Z$ such that $Z_i ≺ Z_j$ if and only if $i < j$
for $i = 1 \ldots, k$ do
    $\Phi \leftarrow Sum\text{-}Product\text{-}Eliminate\text{-}Var(\Phi, Z_i)$
end for
$\phi^* \leftarrow \prod_{\phi \in \Phi} \phi$
return $\phi^*$
end function

function SUM-PRODUCT-ELIMINATE-VAR (Φ, Z)
▷ Φ Set of factors ▷ Z Variable to be eliminated
$\Phi' \leftarrow \{\phi \in \Phi : Z \in Scope[\phi]\}$
$\Phi'' \leftarrow \Phi - \Phi'$
$\psi \leftarrow \prod_{\phi \in \Phi'} \phi$
$\tau \leftarrow \sum_{Z} \psi$
return $\Phi'' \cup \{\tau\}$
end function
From the algorithm it can be seen that the time time complexity for VE for one variable is $O(K^w+1)$, where $w$ is the induced width with respect to an ordering and $K$ is the distinct different values each random variable can take. Hence the overall running time complexity for VE is given by $O(VK^w+1)$, where $w$ is the largest induced width by any variable and $V$ is the number of random variables in the network.

### 2.2 Gibbs Sampling

As we have seen from the section 2.1 VE can also blow up exponentially with dense networks. Hence we need a better set of algorithms to do inference. This is where approximate inference algorithms becomes good.

The general idea behind sampling algorithms is to generate $z', l = 1, ..., L$ independent samples from a probability distribution $p(z)$ and approximate the expectation by taking an average over those generated samples.

In this family of samples we have Gibbs sampling, which is a Markov chain Monte Carlo (MCMC) technique. At first a sample is generated of unobserved variables arbitrarily. Now we iterate over each of the unobserved variables, sample a new value for each variable given the current sample for all other variables [1], refer Algorithm 2.

**Algorithm 2 Generating a Gibbs chain trajectory [1]**

```plaintext
function GIBBS-SAMPLE (X, Φ, P(0)(X), T)
  ▷ X Set of variables to be sampled
  ▷ Φ Set of factors $P_φ$
  ▷ $P(0)(X)$ initial state distribution
  ▷ $T$ Number of time steps
  Sample $x^{(0)}$ from $P(0)(X)$
  for $i = 1, ..., T$ do
    $x^{(t)} ← x^{(t-1)}$
    for each $X_i ∈ X$ do
      Sample $x_i^{(t)}$ from $P_φ(X_i|X_{-i})$
      // Change $X_i$ ∈ $x^{(t)}$
    end for
  end for
  return $x^{(0)}, ..., x^{(T)}$
end function
```
Chapter 3

Experiments and Results

3.1 Setup

For evaluation we generated random BNs of three types, namely linear, fully connected and randomly connected DAG with specified number of vertices and edges. The running time for VE and gibbs sampling is compared for all three networks. The error produced by gibbs sampling is also calculated with respect to VE. All nodes are assigned their conditional probability distributions randomly and similarly is the choice of query variable and evidence. All the random variables used for experiments are discrete and boolean. For random DAG experiments we have taken result by generating 5 different random DAG with same number of nodes and 100 edges. This gives us an average estimate of measures for random DAGs.

3.2 Results

3.2.1 Time Analysis

In this section we present the time taken to run the inference algorithms on all three types of BNs.

Linear BNs

The running time of inference engines are presented in the figure 3.1. As we know that running time of VE is $O(VK^{w+1})$ as discussed in section 2.1. Now as for linear BNs, the induced width of the graph is 1 and for boolean variables $K$ is 2, we have the running time for VE as $O(V2^2) = O(V)$. This makes VE linear with respect to number of nodes. This behaviour is depicted
in the graph too. For Gibbs sampling the performance of the algorithm is
determined by the number of the samples generated. Hence we see that time
taken between 200 samples and 2000 samples generated is linear.

For linear BNs till fairly high number of nodes VE is better as it is linear
in only number of nodes.

**Fully connected BNs**

![Figure 3.1: Running time for inference algorithms in linear BNs](image1)

![Figure 3.2: Running time for inference algorithms in fully connected BNs](image2)
Figure 3.2 shows the running time for VE and Gibbs sampling for fully connected networks. For fully connected networks, the induced width of the graph is $V - 1$, where the $V$ is number of nodes. Thus the running time complexity of VE is $O(V^{2V})$. This makes VE intractable for fully connected networks very quickly. As Gibbs sampling is not affected by the density of the graph it still shows a linear behaviour. After 25 nodes the VE algorithm took too much time to even consider it for comparisons.

**Random DAG**

![Figure 3.3: Running time for inference algorithms for a DAG with 100 edges](image)

The figure 3.3 shows the running time for different inference algorithms for a DAG with 100 random edges. On average the graphs generated had high induced width for most of the cases. Again as VE is exponentially dependent on induced width, the running time blows up again very quickly with relatively low number of nodes. The Gibbs sampling however always produced an estimate in acceptable time bounds.

### 3.2.2 Error Analysis

In the table 3.1 we present error(%) for Gibbs sampling for 2000 samples and 200 samples. The error is calculated as the difference in estimate produced by Gibbs from VE. For analysis purpose we took the readings for the biggest graph in all three different types of BNs. The error is reduced as we increases the number of samples taken.
<table>
<thead>
<tr>
<th>Graph</th>
<th>Error(%) 200 samples</th>
<th>Error(%) 2000 samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>500 node Linear</td>
<td>1.6</td>
<td>0.6</td>
</tr>
<tr>
<td>22 node fully connected</td>
<td>6.3</td>
<td>0.45</td>
</tr>
<tr>
<td>40 node 100 edges DAG</td>
<td>2.362</td>
<td>1.318</td>
</tr>
</tbody>
</table>

Table 3.1: Error in estimate for 2000 samples and 200 samples for various DAGs

![Diagram of Bayesian Network for Mixing time](image)

**Figure 3.4: BN for Mixing time**

### 3.2.3 Mixing Time

For a small random graph of five nodes, we studied the effect of having very skewed probabilities for some nodes like 0.999 or 0.001. The network used is shown in figure 3.4. The tables for probabilities are given in tables 3.2, 3.3, 3.4, 3.5, 3.6. The evidence is taken as $x_4$ and $x_5$ set to true.

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>0.001</td>
</tr>
<tr>
<td>False</td>
<td>0.999</td>
</tr>
</tbody>
</table>

Table 3.2: Probabilty table for $x_1$

The table 3.7 shows the estimation by Gibbs sampler for different number of samples.

The probability estimates for sampling are no way close to correct estimates even after increasing the number of samples by a factor of 10000.
<table>
<thead>
<tr>
<th>$x_2$</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>0.002</td>
</tr>
<tr>
<td>False</td>
<td>0.998</td>
</tr>
</tbody>
</table>

Table 3.3: Probablitiy table for $x_2$

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>0.95</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>0.05</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>0.94</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>0.06</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>0.29</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>0.71</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>0.001</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>0.999</td>
</tr>
</tbody>
</table>

Table 3.4: Probablitiy table for $x_3$

This happens as the probabilities are so highly skewed that those random variables are stuck and Gibbs sampler is not able to change them.
<table>
<thead>
<tr>
<th>$x_3$</th>
<th>$x_4$</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>0.90</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>0.10</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>0.05</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>0.95</td>
</tr>
</tbody>
</table>

Table 3.5: Probability table for $x_4$

<table>
<thead>
<tr>
<th>$x_4$</th>
<th>$x_5$</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>0.70</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>0.30</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>0.01</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>0.99</td>
</tr>
</tbody>
</table>

Table 3.6: Probability table for $x_5$

<table>
<thead>
<tr>
<th>No of Samples</th>
<th>Gibbs Estimate</th>
<th>Error(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0</td>
<td>28.41</td>
</tr>
<tr>
<td>1000</td>
<td>0</td>
<td>28.41</td>
</tr>
<tr>
<td>10000</td>
<td>0.0006</td>
<td>28.357</td>
</tr>
<tr>
<td>100000</td>
<td>0.0009</td>
<td>28.327</td>
</tr>
<tr>
<td>1000000</td>
<td>0.001</td>
<td>28.317</td>
</tr>
</tbody>
</table>

Table 3.7: Gibbs estimate and error for different number of samples for a very skewed DAG
Chapter 4

Conclusion

Variable elimination is an exact algorithm but it becomes intractable very quickly for fully connected and random DAGs. For linear BNs we can use VE as the preferred inference algorithm as it is linear in number of nodes. Gibbs sampling should be the choice of inference algorithm for dense DAGs, which is the case for many real world problems. The error produced by Gibbs decreases as we increase the number of samples generated but so does the running time. So if we are getting an undesired accuracy with less number of samples we can increase the sample size given the running time for higher samples is still acceptable. Gibbs sampling with all its advantages fails almost unavoidably with a network with highly skewed probability distributions as discussed in section 3.2.3. So in this case even increasing the number samples will not work.
Bibliography